LIND(D)UN privacy threat tree catalog

Kim Wuyts
Riccardo Scandariato
Wouter Joosen

Report CW 675, September 2014

KU Leuven
Department of Computer Science
Celestijnenlaan 200A – B-3001 Heverlee (Belgium)

LIND(D)UN privacy threat tree catalog

Kim Wuyts
Riccardo Scandariato
Wouter Joosen

Report CW 675, September 2014

Department of Computer Science, KU Leuven

Abstract

This document contains the updated version of the LIND(D)UN threat tree catalog, which is part of the LIND(D)UN privacy threat modeling methodology. The improvements are the result of a set of extensive empirical studies. A summary is provided for each threat category, followed by a detailed description of each of the corresponding threat trees.

Contents

Introduction

LINDDUN[1] is a privacy threat modeling methodology that aids the analyst in the elicitation of privacy threats. It is a model-based approach, as the methodology requires a data flow diagram (DFD) as representational model of the system to analyze. This DFD will serve as basis for the analysis, as each of its elements will be examined thoroughly for privacy threats. The methodology is also knowledge-based as it provides an overview of the most common attack paths associated with a set of privacy threat categories contained in the acronym LINDDUN (Linkability, Identifiability, Non-repudiation, Detectability, Disclosure of information, Unawareness, Non-compliance). The attack paths are represented as threat trees that detail possible causes of threats that are related to the main threat categories and are specific to a particular DFD element type (entity, data flow, data store, or process).

This document contains the updated version of the LIND(D)UN threat tree catalog. The improvements are based on the results of a set of extensive empirical studies[2].
A summary is provided for each threat category, followed by a detailed description of each of the corresponding threat trees. More information on the methodology can be found on the LIND(D)UN website[3].

[1] Mina Deng, Kim Wuyts, Riccardo Scandariato, Bart Preneel, Wouter Joosen, A privacy threat analysis framework: supporting the elicitation and fulfillment of privacy requirements, Requirements Engineering Journal, volume 16, issue 1, pages 3-32, 2011

[2] Kim Wuyts, Riccardo Scandariato, Wouter Joosen, Empirical evaluation of a privacy-focused threat modeling methodology, The Journal of Systems and Software, volume 96, pages 122-138, October 2014

[3] LINDDUN privacy threat modeling website

Linkability

In general

Being able to sufficiently distinguish whether 2 IOI (items of interest) are linked or not, even WITHOUT knowing the actual identity of the subject of the linkable IOI.

Not being able to hide the link between two or more actions/identities/pieces of information.[1]

Examples are: anonymous letters written by the same person, web page visits by the same user, entries in two databases related to the same person, people related by a friendship link, etc.

Note that often linkability involves IOIs that are linked because they belong to the same subject, however, IOIs can also be linked based on properties (e.g. people visiting the same restaurant, people with a similar disease, etc.).

Consequences

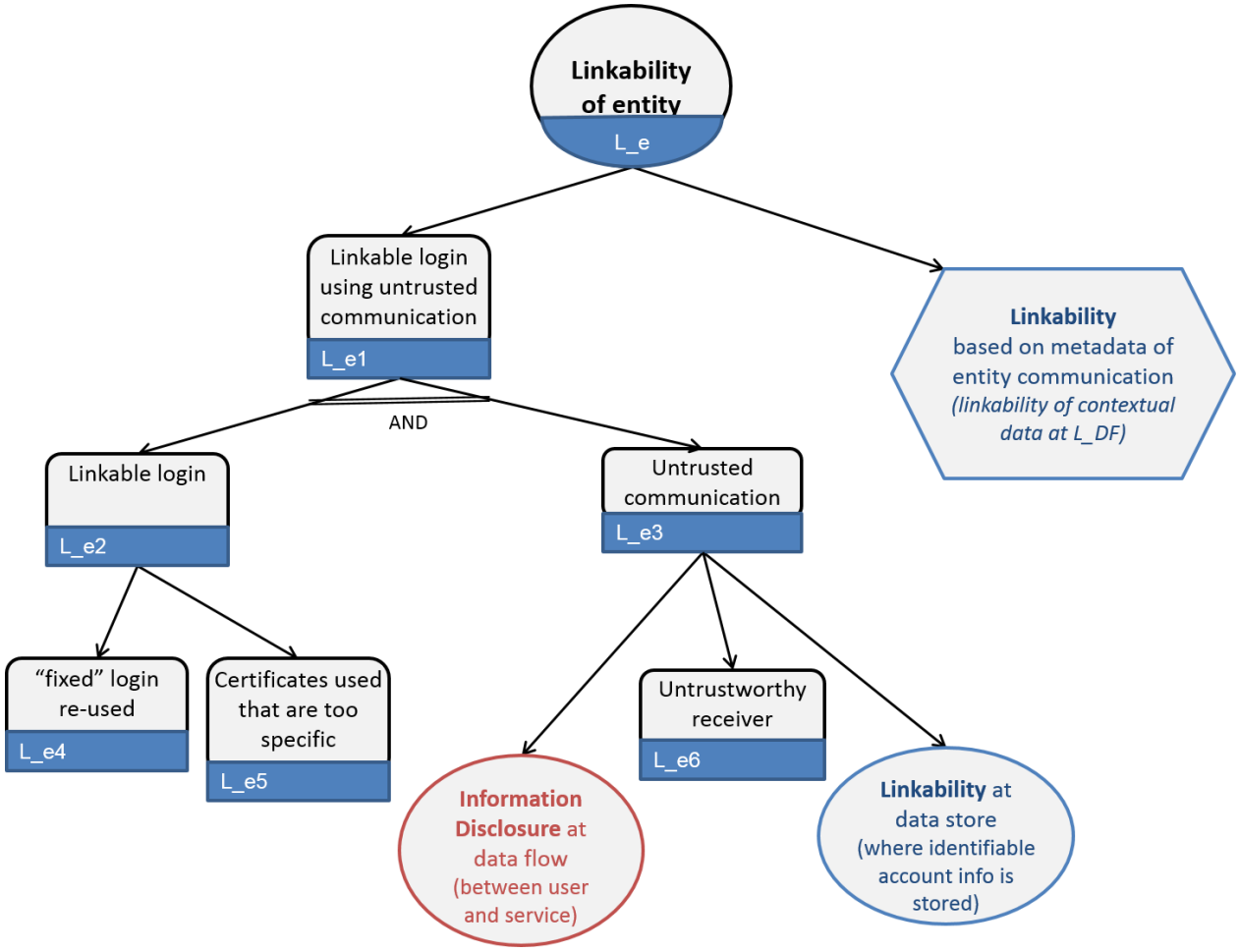
- Can lead to **identifiability** (see Identifiability trees) when too much linkable information is combined
- Can lead to **inference**: when "group data" is linkable, this can lead to societal harm, like discrimination (e.g. if an insurance company knows that people who live in a certain area get sick more often, they might increase their insurance cost for that target group)

Impacted by

- *Data minimization*: the less info is available, the better (see L_DS for more info) (+)
- *Identifiability*: if the subject's identity is known, all related data can obviously be linked (-)

[1] More information about (un)linkability is available in Pfitzmann, Andreas and Hansen, Marit, A terminology for talking about privacy by data minimization: Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management, v0.34, Technical Report, 2010

Linkability of an entity



Tree in general

These threats mainly focus on a subject (the entity) who wants to hide as much of his identifiable information (or at least make it as unlikable as possible). This can occur when the subject wants to authenticate himself to a certain service (multiple authentication principles are shown in the tree), but also during regular communication (browsing, client-server requests, etc.) by means of the contextual information used for the communication.

Leaf nodes explanation

When using a linkable login in combination with untrusted communication, the **entity can become linkable** ***(L_e1)***.

Linkable login (L_e2)

A **linkable login** ***(L_e2)*** can be a **"fixed" login** ***(L_e4)***, like an e-id or a username-password combination, which is being used more than once. As it is being reused, its corresponding IOIs are also linkable based on this login.

Alternatively, using very **detailed certificates** ***(L_e5)*** as means of authentication can lead to linkable logins. For example, certain online services are location-dependent and require users to reside in a certain country. This can be checked in multiple ways. Ideally the certificate (authorized by for example the government or another trusted certificate authority) only proves that the user is indeed a citizen of the required country. However, a more detailed certificate can provide the entire address of the user to prove his residence. As an address is unique (disregarding the fellow residents), the certificate becomes a linkable login.

Untrusted communication (L_e3)

A linkable login will however only lead to linkability of the entity when used over **untrusted communication** ***(L_e3)***

Clearly, when the communication is not secure, the linkable login can be easily **disclosed** ***(ID_DF)***. Note that this information disclosure threat is applicable to *all* communication that transfers the entity's linkable credentials. Usually, this communication is limited to the information flow between the user and service, but when the service transfers the credentials to another service (e.g. a third party authentication service like Facebook Login), obviously also this communication should remain confidential.

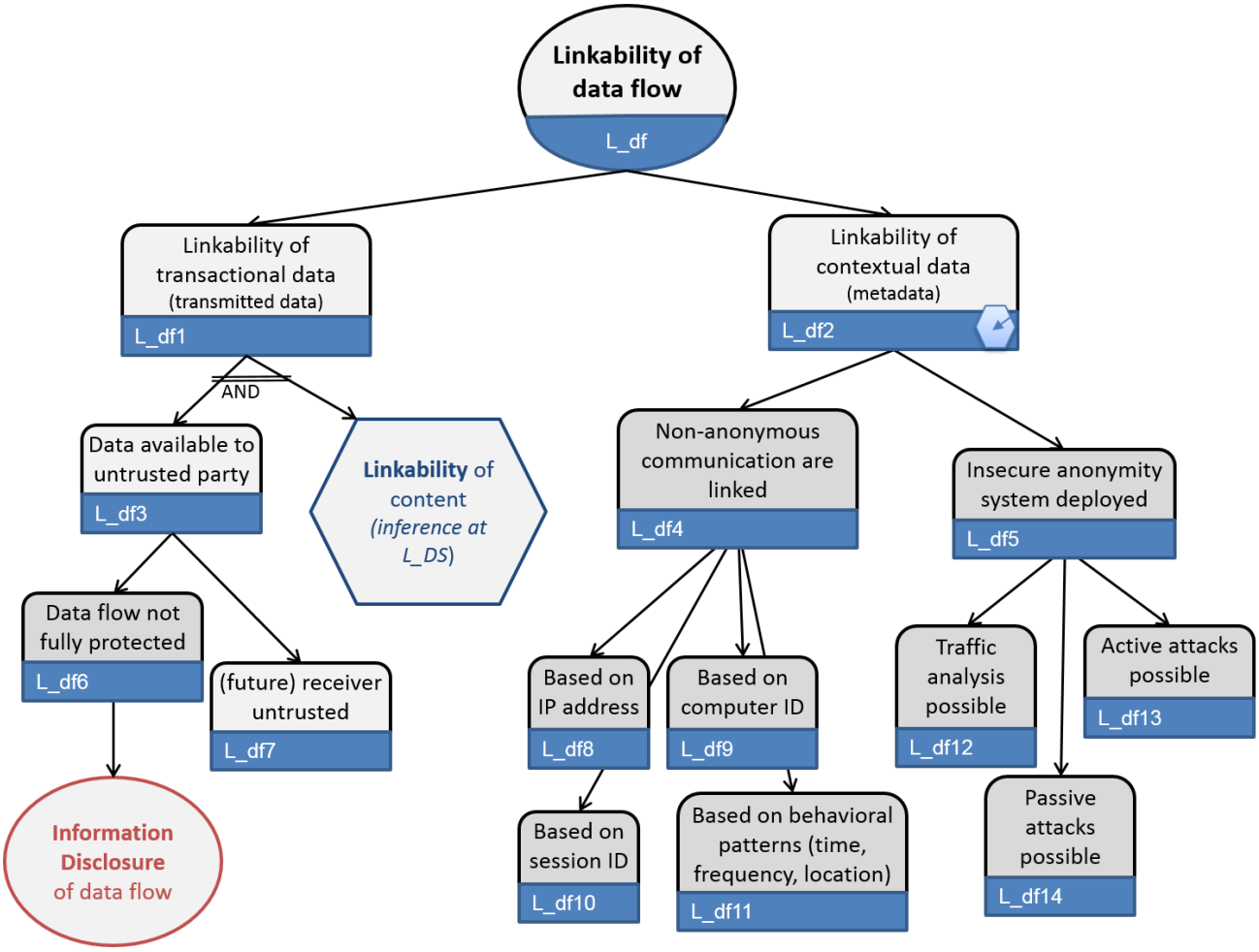
Similarly, **when the receiver of the data is untrustworthy** ***(L_e6)*** (e.g. he fails to anonymize the data during processing or shares the information with other parties), the subject, and all the data he has communicated, become linkable. Note that this receiver can be the service the subject is directly communicating with, but also additional (third party) services that are used by the intended receiver of which the subject might be unaware.

Finally, it is also possible that the receiver is trustworthy and handles the linkable logins in a privacy-friendly fashion by anonymizing them, however, the anonymized **logins** are easily **linkable in the identity management database** (database where internal identifiers are managed and linked to their user accounts)***(L_DS)***. Of course, this is only an issue when this identity management database is not secure.

Linkability base on metadata of entity communication (metadata linkability at L_DF)

Even when no linkable credentials are used, the user can still be linked based on the contextual information of his communication (e.g. based on his IP address or his online behavior) ***(linkability of contextual data at L_DF)***. This threat *only* applies to the communication that is directly connected to the entity, meaning the entity is the sender or receiver of the communication.

Linkability of a dataflow



Tree in general

A distinction should be made between linkability of the contextual data (e.g. IP address necessary for communication) and the transactional data (the actual data that are being communicated).

Contextual data linkability can be resolved by, for example, anonymous routing solutions and should be applied by the sender and/or receiver to protect their own unlinkability.

Transactional data, however, does not necessarily have the sender or receiver as data subject, but can involve a third party subject as well (e.g. when two doctors share information about a patient, this patient is the data subject of the transactional data while he is not involved in the actual communication).

Leaf nodes explanation

Linkability of transactional data (L_DF1)

Transactional data linkability ***(L_DF1)*** can occur when the transmitted data becomes available to an untrusted party ***(L_DF3)***. This either means that the **communication is**

unprotected ***(L_DF6)*** and hence **information disclosure threats** ***(ID_DF)*** apply, or that a **receiver cannot be trusted*****(L_DF7)***. This receiver can be the direct receiver of the information, or can be a "future" receiver when the receiving party communicates the transactional data to other parties (e.g. when the accounting is being outsourced to a third party).

Sharing data with untrusted parties will however only lead to linkability when the **information** that is made available is actually **linkable** ***(insufficient minimization inference of L_DS)***. This mainly occurs when the data that are being sent are not being minimized sufficiently. When protecting a user's privacy ideally only the minimal set of information should be provided.

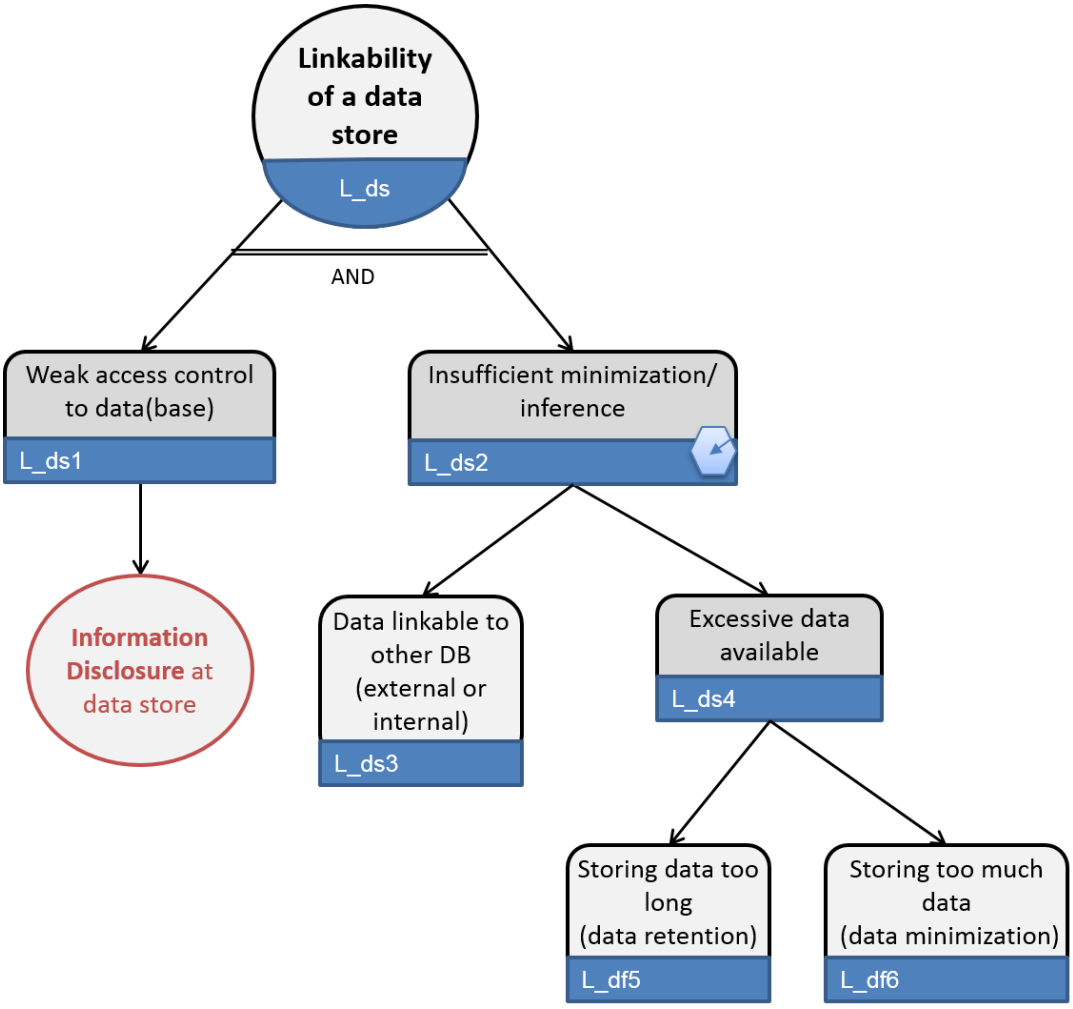
Linkability of contextual data (L_DF2)

Contextual data becomes linkable ***(L_DF2)*** when **non-anonymous communication** ***(L_DF4)*** is used. The data flow can be for example linked on **IP address** ***(L_DF8)***, **computer iD** ***(L_DF9)***, **session ID** ***(L_DF10)***, or even based on **certain patterns** ***(L_DF11)*** (like time, frequency, and location or browser setting, etc.).

Alternatively, it is possible that the **anonymity system that is being used is insecure** ***(L_DF5)***. This can enables **traffic analysis** ***(L_DF12)*** to extract information out of patterns of traffic; **passive attacks** ***(L_DF14)***, like long-term intersection attacks, traffic correlation and confirmation, fingerprinting, epistemic attacks (route selection), and predecessor attacks; or **active attacks*****(L_DF13)***, like N-1 attacks, Sybil attack, traffic watermarking, tagging attack, replay, and DoS attack. More information about these attacks can be found in [1].

[1] G. Danezis, C. Diaz, and P. Syverson, Systems for Anonymous Communication, in CRC Handbook of Financial Cryptography and Security, p. 61. Chapman & Hall, 2009.

Linkability of a data store



Tree in general

Linkability in a data store occurs when one has access to the data store and when insufficient data minimization is applied. This means that too much data are being stored which enables a large set of information that can be used (e.g. by data miners) to look for links.

The most obvious consequence of linking lots of information is that more pseudo-identifiers are linked which can result in ***identifiability*** (e.g. knowing one's city, gender, age or even first name does not reveal an identity, but when combined the anonymity set suddenly becomes a lot smaller and can already lead to identification, depending on the city's population size and the uniqueness of the person's first name. Thus the more data available and linkable (based on (pseudo)identifiers), the more likely the chance of identification.

Another result of linkability is ***inference***. Instead of linking data that belongs to the same person, data are linked based on certain properties to deduce relationships between them and generalize them. This can be used in a rather innocent fashion to determine the best way to organize groceries in a grocery store (e.g. people who buy hamburgers usually buy

buns at the same time, hence they are stored close to each other). This inference can however also have a more judgmental nature if it is used to discriminate a certain population (e.g. people living in a certain neighborhood have a higher chance of cancer, hence their health insurance fee is higher than the surrounding cities). Inference can thus lead to societal harm.

Leaf nodes explanation

Weak access control (L_DS1)

A requirement for linkability in the data store is **weak access control to the data store** ***(L_DS1)***. This can occur when the data are not stored in a confidential fashion and hence **information disclosure threats** exist ***(ID_DS)***.

Insufficient minimization/Inference (L_DS2)

When having access to the data store, linkability becomes a threat when data is **insufficiently minimized** ***(L_DS2)*** [1]. The more information is available the higher the chance of **inference**, which is a key element of data mining that derives ideas and conclusions by combining (linkable) information.

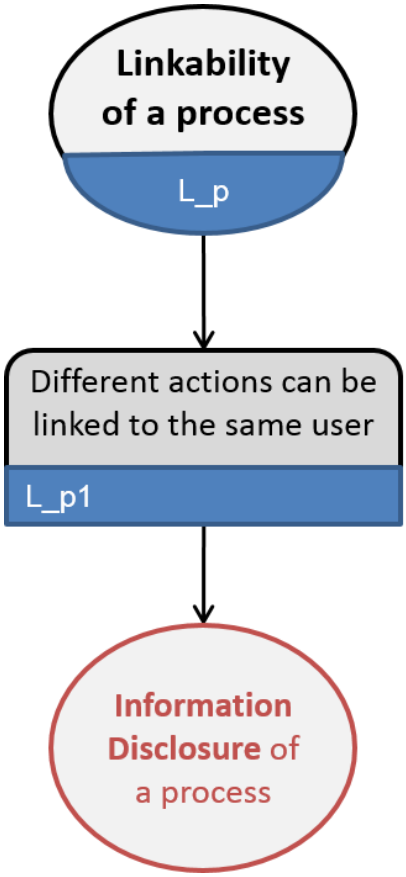
The data can be **linked to data in another database** ***(L_DS3)***. This other database can be both internal to the system that is being analyzed or external (data from a partner company, public data online, like Facebook data, etc.)[2].

Or, simply **too much data are available** ***(L_DS4)*** in the data store that is being analyzed. This can occur when data are **stored too long** ***(L_DS5)*** instead of removing them when no longer needed (and thus resulting in too much information) or **storing more information than required** ***(L_DF6)*** for the purpose of collection (e.g. storing a subject's entire address when only his city is required). Both data retention and data minimization threats originate from the data protection legislation principles.

[1] Note that this minimization branch can also be considered from a wider perspective, when one does not (only) focus on data minimization, but also minimization of central storage, of risk, of trust, etc. This tree does however only discuss minimization of data, as it is the main privacy concern regarding privacy.

[2] When this additional database is an identity management database that stores account data (and the login is identifiable), linking the IOIs to this database will result in identifiability.

Linkability of a process



Tree in general

The threat tree of **linkability of process** suggests that the only way to prevent different actions being linked to the same subject is by gaining access to the process. Note that this threat is very rare.

Leaf nodes explanation

Linkability of a process can only occur after **information disclosure of this process** ***(ID_P)***. We therefore refer to that tree.

Identifiability

In general

Being able to sufficiently identify the subject within a set of subjects (i.e. the anonymity set).

Not being able to hide the link between the identity and the IOI (an action or piece of information) [1].

Examples are: identifying the reader of a web page, the sender of an email, the person to whom an entry in a database relates, etc.

Consequences

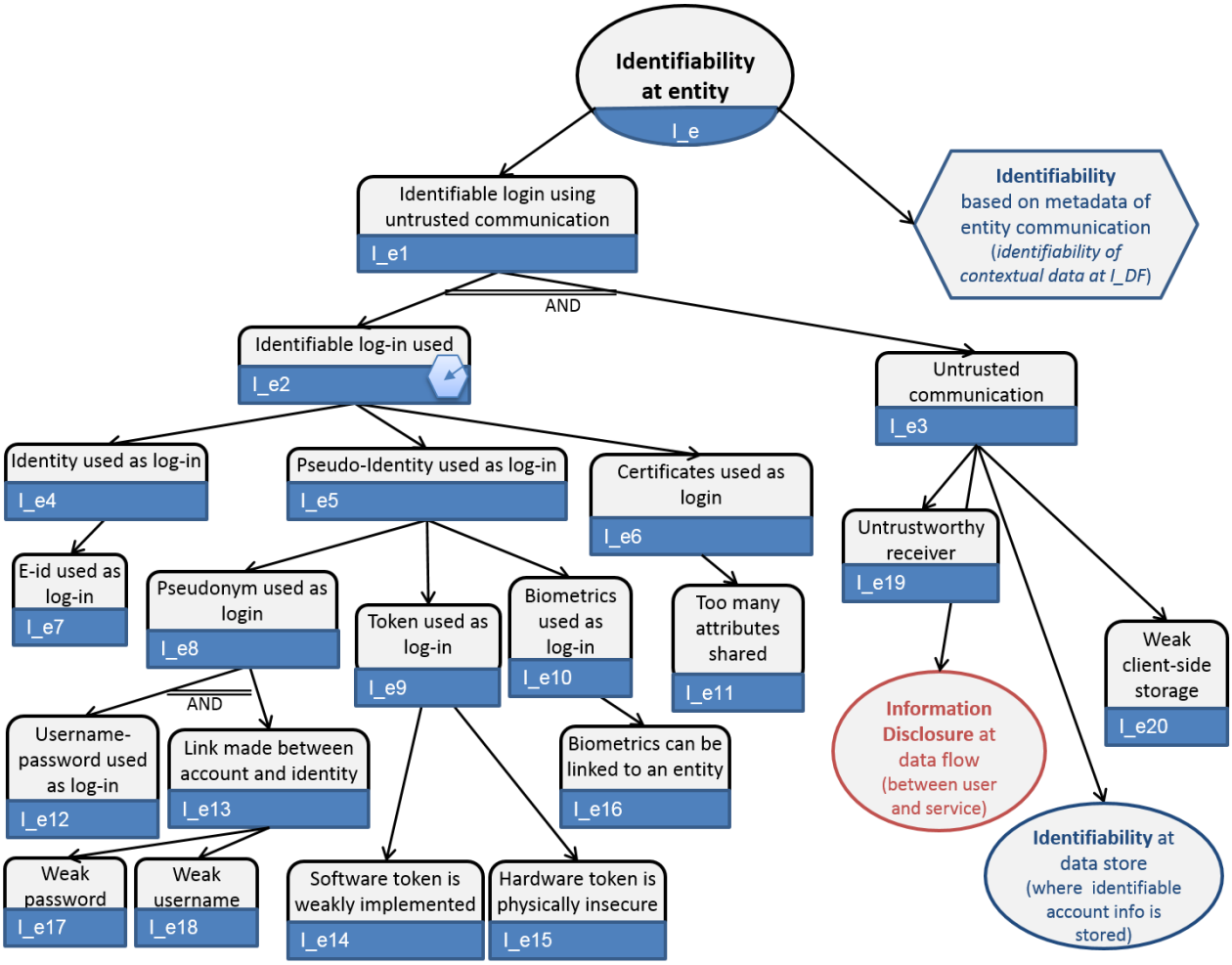
- Can lead to severe privacy violations (when subject assumes he is anonymous)

Impacted by

- *Data minimization*: the less info is available, the better (see L_DS for more info) (+)
- *Linkability*: the more information is linked, the higher the chance the combined data are identifiable (the more attributes are known, the smaller the anonymity set) (-)

[1] More information about anonymity and pseudonymity is available in Pfitzmann, Andreas and Hansen, Marit, A terminology for talking about privacy by data minimization: Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management, v0.34, Technical Report, 2010

Identifiability of entity



Tree in general

These threats mainly focus on a subject (the entity) who wants to hide as much of his identifiable information as possible. This can occur when the subject wants to authenticate himself to a certain service (multiple authentication principles are shown in the tree), but also when (anonymously) browsing by means of the contextual information using for the communication.

Leaf nodes explanation

When an **identifiable login is used and is communicated in an untrustworthy matter**, the entity can be identifiable ***(I_e1)***.

Identifiable login used (I_e2)

Several types of identifiable logins exist. The most obvious is the **e-id** ***(I_e7)***, which means using your **real identity** ***(I_e4)***.

Alternatively, a **pseudo-identity** ***(I_e5)*** can be used. The most common pseudo-identity is a **pseudonym** ***(I_e8)***, using a username-password combination. Although in theory this can

provide anonymity (you can choose an unrelated username and password, or it can be assigned to you), in practice this pseudonym is often not very anonymous. Either the **username can be easily linked** *(I_e18)* to the user's identity (e.g. the user's firstname and/or lastname) or even **the password** *(I_e17)* can contain identifiable information (people tend to use easy to remember password like their birthday). Another pseudo-identifier can be a **token** *(I_e9)*, which can be either a **hardware** *(I_e15)* or **software** *(I_e14)* token [1] (e.g. a smartcard, usb token, a disconnected token, a file, etc.). When the token is badly designed (either physically or implementation-wise), the entity can be identified. A final type of pseudo-identity is **biometrics** *(I_e10)* (e.g. fingerprints) that are identifiable when **the biometrics themselves can be linked back to the actual identity** *(I_e16)*.

Another type of identifiers are **certificates** *(I_e6)*. They are the most privacy-friendly authentication type as they only aim at proving certain properties about the entity (e.g. older than 18, living in the US, female, etc.). The entity can however still become identifiable when **the certificate contains too much (precise) properties** *(L_e11)*. The more specific a certificate is (and thus unique), the more identifiable it becomes.

Untrusted communication (I_e3)

An identifiable login can lead to identifiability of an entity when this login is used over **untrusted communication** *(I_e3)*. This subtree is actually very similar to the untrusted communication subtree of linkability (L_e3) as the threats are closely related.

When the communication is not secure, the identifiable login can be **easily disclosed** *(ID_DF)*. This information disclosure threat is applicable to all communication that transfers the entity's identifiable credentials. This communication is not necessarily limited to the information flow between the user and service, as the service can decide to transfers the credentials to another service (e.g. a third party authentication service like Facebook Login). Obviously, also this communication should remain confidential.

Similarly, **when the receiver of the data is untrustworthy** *(I_e6)* (e.g. he fails to anonymize the data during processing or shares the information with other parties), the subject, and all the data he has communicated, become identifiable. Note that this receiver can be the service the subject is directly communicating with, but also additional (third party) services that are used by the intended receiver of which the subject might be unaware.

Also, **when the user is not careful when storing his credentials** *(I_e20)* (e.g. writing username and password on a paper near the computer, failing to install security measures which allow keyloggers or other kinds of eavesdropping, etc.), the identifiable login can be easily intercepted and hence the entity becomes identifiable.

Finally, even when the receiver is trustworthy and stores the identifiable user credentials in a privacy-friendly fashion by anonymizing them, the entity can become identifiable when the **anonymized logins are identifiable in the identity management database** *(I_DS)* if it is not properly secured (e.g. having an identity management database where the full credentials

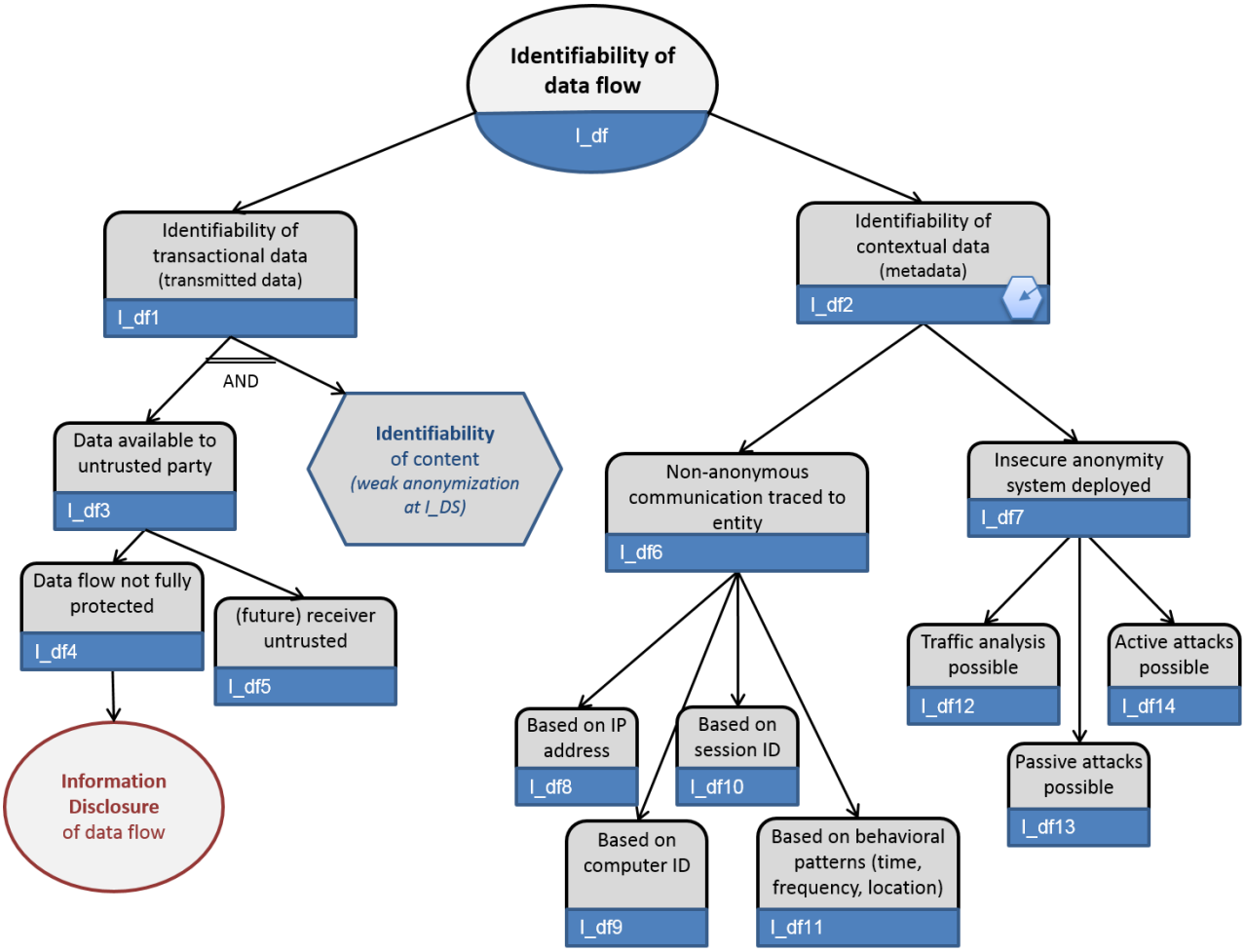
are stored, and an internal (anonymized) identifier for the main data; however, the IDM database is not encrypted).

Identifiability based on metadata of entity communication (identifiability of contextual data at I_DF)

Even when no identifiable credentials are used, the user can still be **identified based on the contextual information of his communication** (e.g. based on his IP address or his online behavior) ***(identifiability of contextual data at I_DF)***. This threat only applies to the communication that is directly connected to the entity, meaning the entity is the sender or receiver of the communication.

[1] Wikipedia provides a nice overview of the different kinds of security tokens

Identifiability of data flow



Tree in general

A distinction should be made between identifiability of the contextual data (e.g. IP address necessary for communication) and the transactional data (the actual data that are being communicated).

Contextual data identifiability can be resolved by, for example, anonymous routing solutions and should be applied by the sender and/or receiver to protect their own anonymity.

Transactional data, however, does not necessarily have the sender or receiver as data subject, but can involve a third party subject as well (e.g. when two doctors share information about a patient, this patient is the data subject of the transactional data while he is not involved in the actual communication). Transactional data identifiability can occur when the flow is unprotected and hence information disclosure threats apply, and, when this disclosed information itself is identifiable. Another threat to transactional data identifiability exists when the data is being sent to an untrustworthy receiver (or future receiver). Transactional data identifiability should be resolved at the origin of the data or at

least before the data cross a trust boundary (which can thus be much earlier in the flow than the current sender).

Leaf nodes explanation

Identifiability of transactional data (I_DF1)

Transactional data identifiability ***(L_DF1)*** can occur when the transmitted data becomes available to an **untrusted party** ***(I_DF3)***. This either means that the **communication is unprotected*****(I_DF4)*** and hence **information disclosure threats** ***(ID_DF)*** apply, or that a **receiver cannot be trusted** ***(I_DF5)***. This receiver can be the direct receiver of the information, or can be a "future" receiver when the receiving party communicates the transactional data to other parties (e.g. when the accounting is being outsourced to a third party).

Sharing data with untrusted parties will however only lead to linkability when the **information** that is made available is actually **linkable** ***(insufficient minimization of I_DS)***. This mainly occurs when the data that are being sent are not being minimized sufficiently. When protecting a user's privacy ideally only the minimal set of information should be provided.

Identifiability of contextual data (I_DF2)

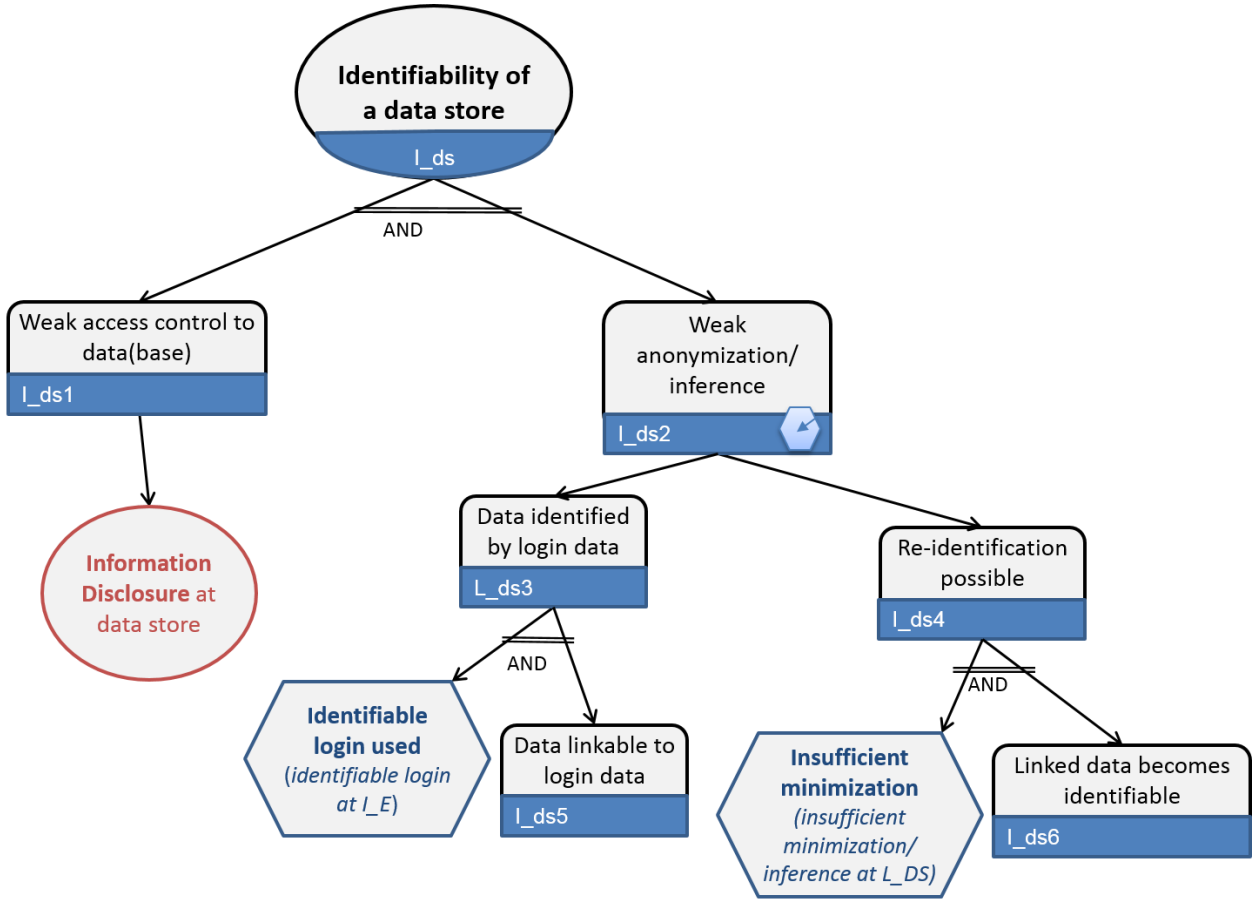
This subtree is actually identical to the *subtree of linkability (L_DF2)*. The summary of this subtree will thus be similar as well. Note however that is, generally speaking, easier to link data flows based on their contextual information, than actually identify (e.g. knowing that a certain user visits website X every day at 8PM will make his actions linkable based on this time pattern, however, this does not provide sufficient information to actually identify him).

Contextual data becomes identifiability ***(I_DF2)*** when **non-anonymous communication*****(I_DF6)*** is used. The data flow can be for example linked on **IP address** ***(I_DF8)***, **computer iD*****(I_DF9)***, **session ID** ***(I_DF10)***, or even based on **certain patterns** ***(I_DF11)*** (like time, frequency, and location or browser setting, etc.).

Alternatively, it is possible that the **anonymity system that is being used in insecure** ***(I_DF7)***. This can enables **traffic analysis** ***(I_DF12)*** to extract information out of patterns of traffic; **passive attacks** ***(I_DF13)***, like long-term intersection attacks, traffic correlation and confirmation, fingerprinting, epistemic attacks (route selection), and predecessor attacks; or **active attacks*****(I_DF14)***, like N-1 attacks, Sybil attack, traffic watermarking, tagging attack, replay, and DoS attack. More information about these attacks can be found in the work of Danezis, Diaz, and Syverson[1].

[1] G. Danezis, C. Diaz, and P. Syverson, Systems for Anonymous Communication, in CRC Handbook of Financial Cryptography and Security, p. 61. Chapman & Hall, 2009.

Identifiability of data store



Tree in general

Identifiability at a data store can only occur when the data store itself is not sufficiently protected.

When the data store can be accessed and the data are insufficiently anonymized, the data can become identifiable. Either because the data are identified because they are linked to (identifiable) login data, or because the data are re-identified by lack of (sufficient) data minimization.

Leaf nodes explanation

Weak access control (I_DS1)

In order to have identifiability at the data store, one needs access to it. Thus **weak access control*(I_DS1)*** is a prerequisite, which is possible when there is **information disclosure at the data store*(ID_DS)***.

Weak anonymization/ inference (I_DS2)

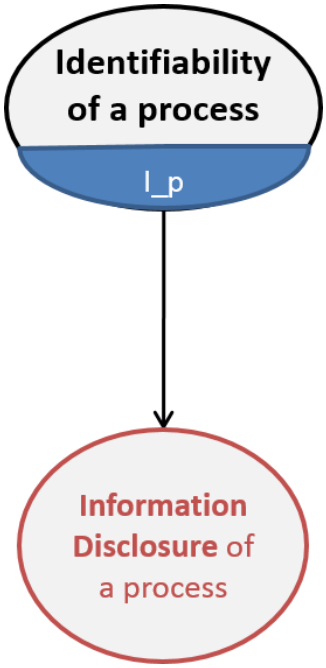
When access to the data store is provided, data become identifiable when there is **weak data anonymization** . (and/or when inference techniques are applied) ***(I_DS2)***. This can

occur in two ways. Either **the data are identified by means of login data** ***(I_DS3)*** when **the data are linkable to login data** ***(I_DS5)*** [1] and these **login data are identifiable** ***(identifiable login used of I_E)***.

Or, the **data can be re-identified** ***(I_DS4)*** because they are **insufficiently minimized*****(insufficient minimization at L_DS)*** and given the abundance of **linked data**, they **become identifiable** ***(I_DS6)***. Clearly, the more information is available, the more unique it becomes and hence the smaller the anonymity set.

[1] This node is actually a specific case of the L_DS3 leaf node (data linkable to other database). When the other database contains identifiable data, like an identity management database does, the entire dataset becomes identifiable

Identifiability of a Process



Tree in general

The threat tree of **Identifiability of process** suggests that the only way to prevent different actions being linked to the same subject is by gaining access to the process. Note that this threat is very rare.

Leaf nodes explanation

Identifiability of a process can only occur after **information disclosure of this process** ***(ID_P)***. We therefore refer to that tree.

Non-repudiation

In general

Not being able to deny a claim.

The attacker can thus prove a user knows, has done or has said something. He can gather evidence to counter the claims of the repudiating party.

Typical non-repudiation examples exist in the context of anonymous online voting systems, and whistleblowing systems where plausible deniability is required.

Note that this threat is actually a security goal, as for certain systems (e.g. systems where payments are involved) non-repudiation is an important asset. This should however not result in any conflicts, as systems that require non-repudiation as a security goal, will not need plausible deniability for the same data/ subsystem.

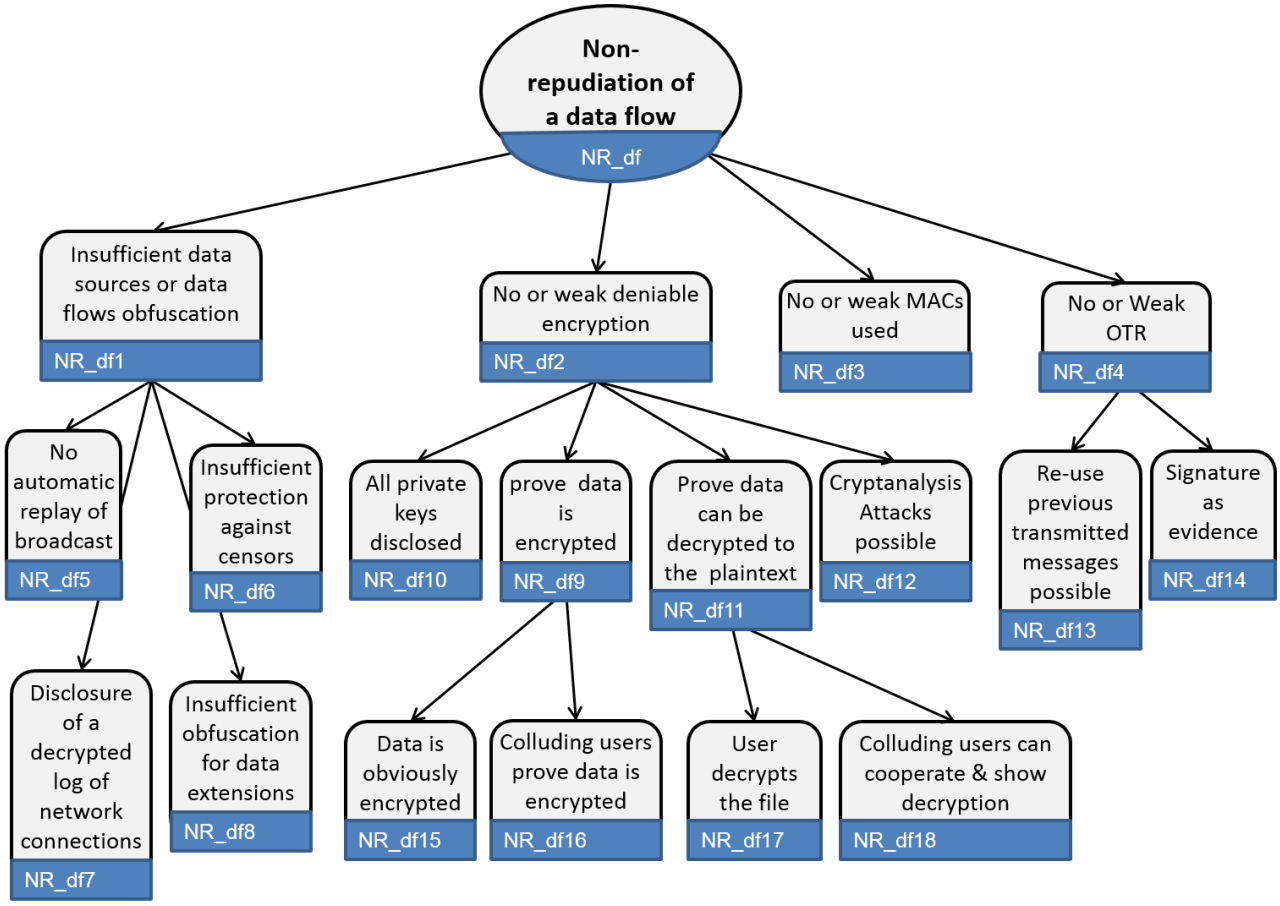
Consequences

- *Accountability*: when a person is not able to repudiate an action or piece of information, he can be held accountable. (e.g. a whistleblower can be prosecuted)

Impacted by

- *Identifiability*: if data are identifiable, it will be hard to repudiate (-)

Non-repudiation of data flow



Tree in general

Non-repudiation of a data flow implies that the subject cannot deny having sent a message. This can occur when data sources of flows are insufficiently obfuscated, when weak deniable encryption, weak MACs, or weak off-the-record messaging are used.

Leaf nodes explanation

Four types of threats can lead to non-repudiation of data flow.

Insufficient data sources or data flows obfuscation (NR_DF1)

The first type is insufficient obfuscation for data sources or data flows, which means that the attacker can gain access to at least part of the data flow or data source. This can occur in a number of cases, for example, there is **no automatic replay of broadcasts** ***(NR_DF5)***, such that the sender of a file is sufficiently distinguishable from those who are merely relaying it. Another example is when a complete **decrypted log of all network connections** to and from a user's computer is **disclosed** ***(NR_DF7)***, resulting in the disclosure of the origin of data flow. The final examples are that there is **insufficient protection against censors** ***(NR_DF6)*** or **insufficient obfuscation of data extensions** ***(NR_DF8)***, such that operators or users of the network are able to know where the data comes from.

No or weak deniable encryption (NR_DF2)

The second type of threat is that little or a weak deniable encryption technique is used to protect data flow. One possible attack path is to **prove data is encrypted** *(NR_DF9)*, either the encrypter proves the **data is obviously an encryption** *(NR_DF15)* or **colluding users prove together that the data is encrypted** *(NR_16)***.**

The second attack path is to prove data can be **decrypted to a valid plain text** *(NR_DF11)*, which can occur when the encrypter **decrypts the file** *(NR_DF17)* or **colluding users can cooperate** and show the decrypted message *(NR_DF18)*.

The third attack path shows that all **private keys are disclosed** *(NR_DF10)*, which clearly enables decryption.

Finally, also **cryptanalysis** *(NR_DF12)* is possible to attack the used encryption scheme.

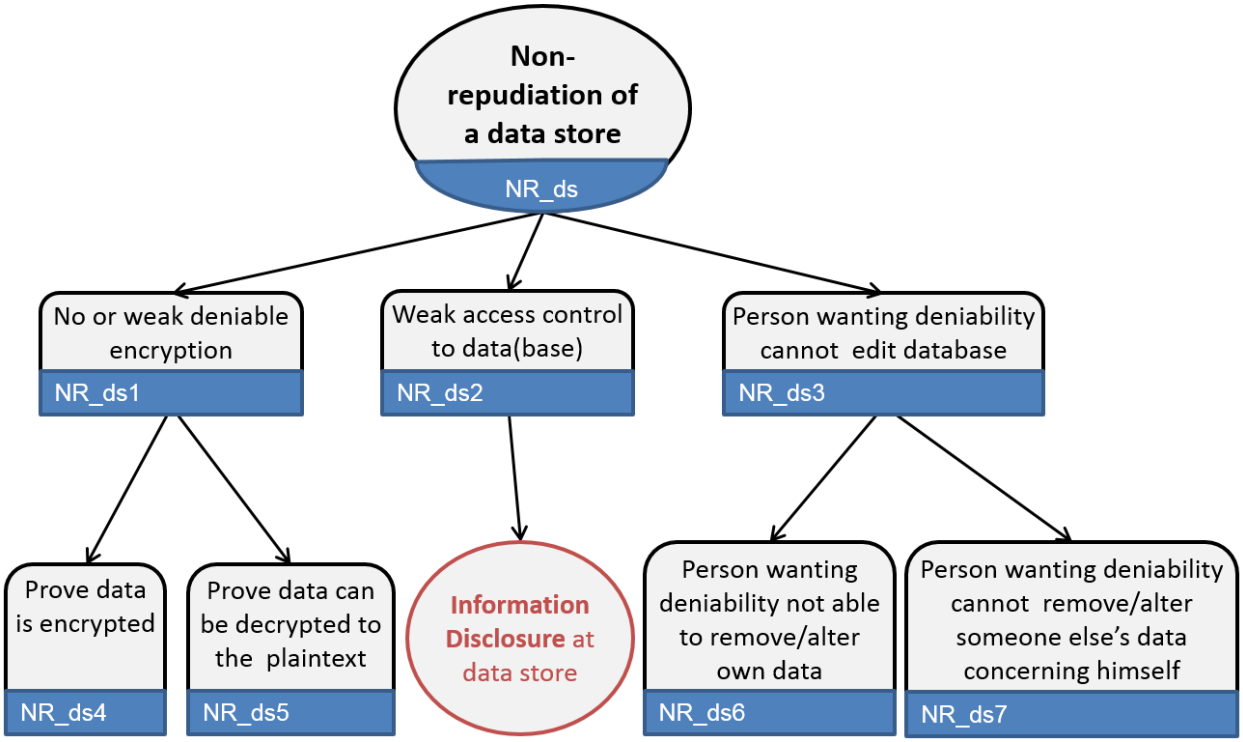
No or weak MACs used (NR_DF3)

The third type of threat is **the lack of strong message authentication codes (MAC)** *(NR_DF3)* to ensure integrity of data flow content, such that an attacker can forge authentic looking messages and pretend that a certain data flow comes from a subject.

No or weak off-the-record messaging (NR_DF4)

The final precondition indicates that there is **little or a weak Off-the-Record Messaging (OTR)***(NR_DF4)* used, such that in a conversation it is not possible to provide both deniability for the conversation participants and confidentiality of conversations content at the same time. Possible attack paths include **replaying of previous transferred messages** *(NR_DF13)*, and the **use of signatures** *(NR_DF14)* to demonstrate communication events, participants and communication content.

Non-repudiation of data store



Tree in general

Non-repudiation in the data store refers to the threat where a subject is not able to deny certain data in the database. This can be either data he has stored himself or data that others have stored about the subject.

Leaf nodes explanation

No or weak deniable encryption (NR_DS1)

When **little or a weak deniable encryption** ***(NR_DS1)*** is used to protect the data, it can be **proven that data are an encryption** ***(NR_DS4)*** or **can be decrypted to a valid plaintext*****(NR_DS5)***.

Weak access control to the database (NR_DS2)

When there is **weak access control to the database** ***(NR_DS2)***, the stored data are no longer deniable. This can occur when there is a threat of **information disclosure at the data store*****(ID_DS)***.

Person wanting deniability cannot edit database(NR_DS3)

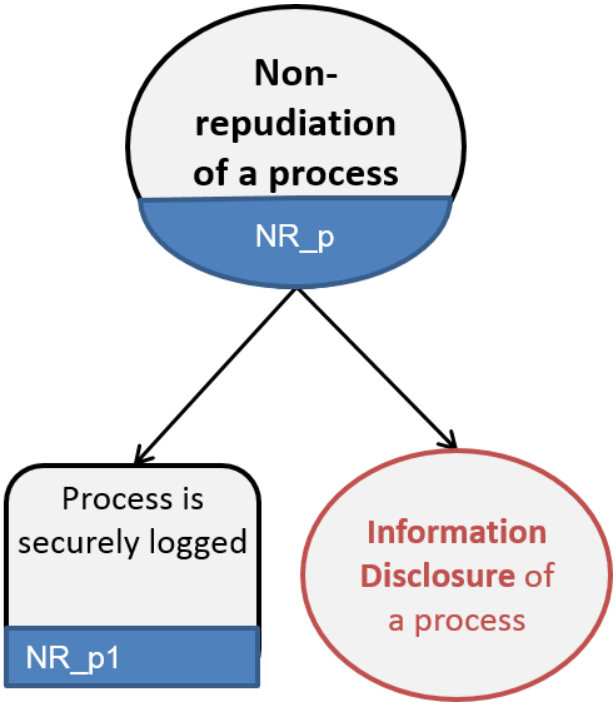
If subjects want deniability but are **not able to edit data in the database *(NR_DS3)*** to cover their tracks, their data becomes non-reputable. It can be either **impossible to remove or alter the user's own data *(NR_DS6)*** or **impossible to remove or alter someone else's data concerning the user himself *(NR_DS7)***.

Note that this threat does not apply solely to a database that can be directly accessed by the user. It applies to all collected data.
For example, Google has to implement the 'right to be forgotten' which allows data subjects to request removal of personal information from Google's search index if the links are inadequate, irrelevant or no longer relevant, or excessive in relation to the purposes for which they were processed [1] [2].

[1] Factsheet of European Commission on the right to be forgotten

[2] This is actually an extension of the unawareness threat regarding data reviewal (*U_5* in the unawareness tree). Not only should a subject be aware of the data that is collected about him (U), he should also be able to revoke it (NR).

Non-repudiation of a process



Tree in general

Non-repudiation of a process implies that it cannot be denied that the process has been run. It is however a very rare threat.

Leaf nodes explanation

Non-repudiation of a process can be achieved in two ways.

Either the process loses its confidentiality and **information disclosure attacks at the process*(ID_P)*** are possible, or the process uses a **secure log *(NR_P1)*** to create an overview of all actions, which can evidently be traced back to the user.

Detectability

In general

Being able to sufficiently distinguish whether an item of interest (IOI) exists or not.

Note that detectability does not imply information disclosure. Detectability concerns IOIs of which the content is not known (to the attacker).

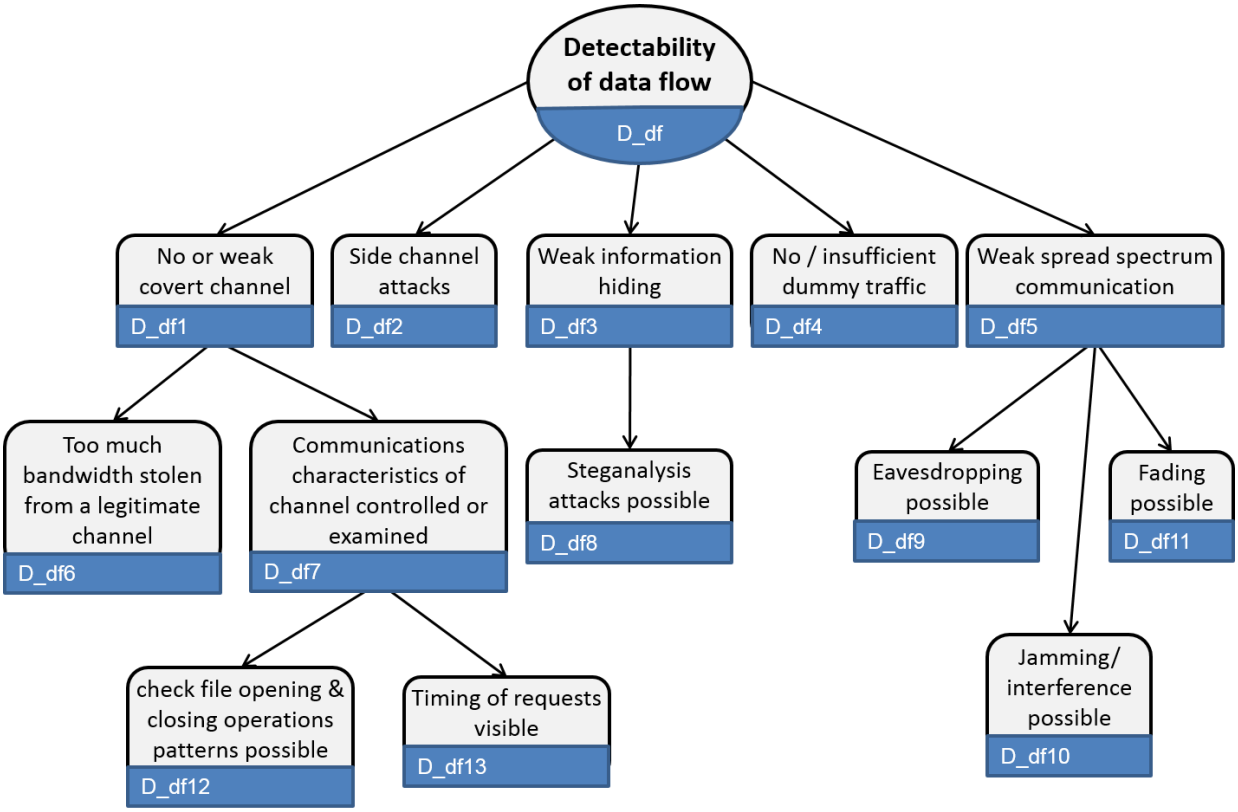
Consequences

- *Inference*: by detecting whether an IOI exists, one can deduce certain information, even without actually having access to that information (e.g. by knowing that a celebrity has a health record in a rehab facility, you can deduce the celebrity has an addiction, even without having access to the actual health record)

Impacted by

- /

Detectability of a dataflow



Tree in general

Knowing that a message is sent, without actually knowing what is contained in the message, can often reveal additional (sensitive) information. For example, when a smart grid system only sends consumption messages from the customer's home system to the back-end when electricity is being consumed, detecting that such a message [1] is sent to the back-end reveals that there are currently people in the house.

Leaf nodes explanation

No or weak covert channel (D_DF1)

A first type of threat that can lead to detectability of a data flow is that the **system lacks a covert channel** ***(D_DF1)***. This can happen when the covert channel uses **too much bandwidth from a legitimate channel** ***(D_DF6)***, resulting in the detection of the covert communication. It can also be because the patterns or **characteristics of the communications medium** of the legitimate channel are controlled or examined by legitimate users ***(D_DF7)***, e.g. checking **file opening and closing operations patterns** ***(D_DF12)*** or watching the **timing of requests** ***(D_DF13)***, such that covert communication is detected.

Side channel attacks (D_DF2)

Side channel analysis ***(D_DF2)*** can be based on timing information, power consumption, electromagnetic leaks, etc. It is used as an extra source of information which can be exploited to detect the communication.

Weak information hiding (D_DF3)

When **weak information hiding techniques** ***(D_DF3)*** are used, **steganalysis attacks** ***(D_DF8)*** are possible (detecting messages hidden using steganography).

No or insufficient dummy traffic (D_DF4)

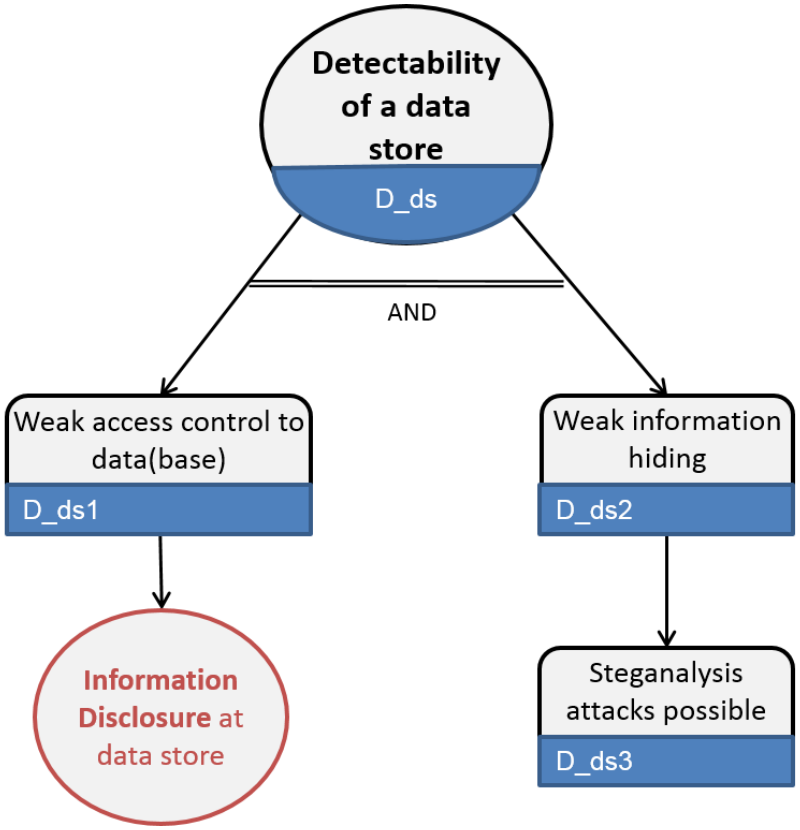
Transmitted data can become detectable when there is **no or insufficient dummy traffic** ***(D_DF4)***sent at some lower layer of the communication network, such that messages fail to appear random for all parties except the sender and the recipient(s).

Weak spread spectrum communication (D_DF5)

The detectability threat can occur because of a **weak spread spectrum communication*****(D_DF5)***, resulting in deficiencies in the establishment of secure communications (allowing **eavesdropping** ***(D_DF9)***), insufficient resistance to natural **interference and jamming*****(D_DF10)***, and insufficient resistance to **fading** ***(D_DF11)***.

[1] In practice, certain household appliances will still be consuming electricity, even though nobody is home. Therefore, detecting messages being sent at irregular or very short intervals, will reveal that someone is home.

Detectability of a Data store



Tree in general

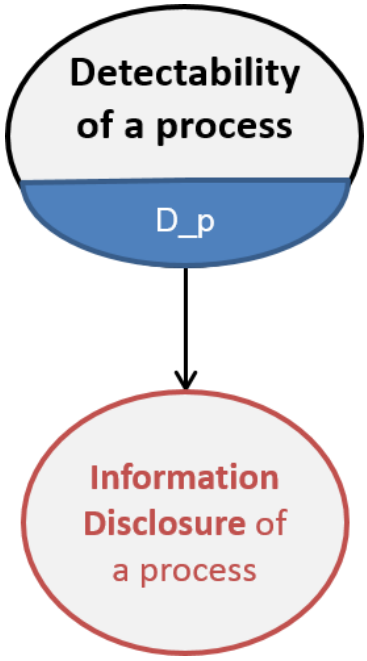
Knowing that an IOI exists, without actually having access to it, can already reveal (possibly sensitive) information. For example, knowing that a rehab clinic has a file on a certain celebrity, already reveals information (i.e. the celebrity has been in rehab), without actually having access to the file.

Leaf nodes explanation

Detectability at the data store can occur when there is **insufficient access control** ***(D_DS1)***, because of **information disclosure threats** ***(ID_DS)*** and if **insufficient information hiding techniques** ***(D_DS2)*** are applied, such that information is revealed due to weak steganography algorithms which enable **steganalysis attacks** ***(D_DS3)***.

Note that the access control should not only apply to the actual data but also to their corresponding metadata, as knowing that an item exists without actually having access to it, can also reveal sensitive information.

Detectability of a Process



Tree in general

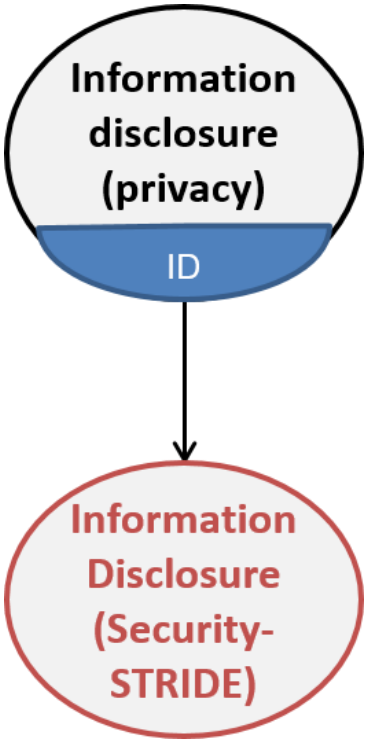
Detectability of process implies that it can be detected that the process has been run. It is however a very rare threat.

Leaf nodes explanation

Identifiability of a process can only occur after Similar to the other privacy threats related to a process, the detectability of process threat also refers to the threat of **information disclosure of this process** ***(ID_P)***. We therefore refer to that tree.

Information Disclosure

Note that the information disclosure trees are not part of LINDDUN, but of Microsoft's STRIDE. As privacy depends on security, LINDDUN also includes STRIDE's Information Disclosure threats (which, on their turn, refer to other STRIDE trees).
It is however advised to execute a full security analysis in advance or in parallel with LINDDUN.



Tree in general

The threat tree concerning *information disclosure of data flow, data store, and process* refers to the security threat tree of information disclosure. This illustrates the fact that privacy properties are part of security properties, and privacy may depend on security. Please access the corresponding STRIDE threat trees through the links in the second navigation bar on the left.

Unwareness

In general

Being unaware of the consequences of sharing information.

Often users are not aware of the impact of sharing data. This can be data shared to friends on facebook, but also personal information shared with other services (i.e. loyalty cards, online services, …)

Unawareness threats differ from data minimization threats (L_DS) in the sense that concerning data minimization it is the responsibility of the (back-end) system to minimize the data that are being stored, while for unawareness it is the provider of the data himself who is responsible and should be aware of the consequences of sharing (too much) information. Nevertheless, the system itself can support the users in making privacy-aware decisions.

Ideally, all users (data providers) should be clearly informed and educated of the consequences of sharing data using (online) services. Our analysis can however not impact the entire society, hence these threats will only focus on the provisions the system itself can take to guide and educate the user concerning his data sharing.

This threat **only applies to an entity**, as other DFD elements do not input additional information in the system.

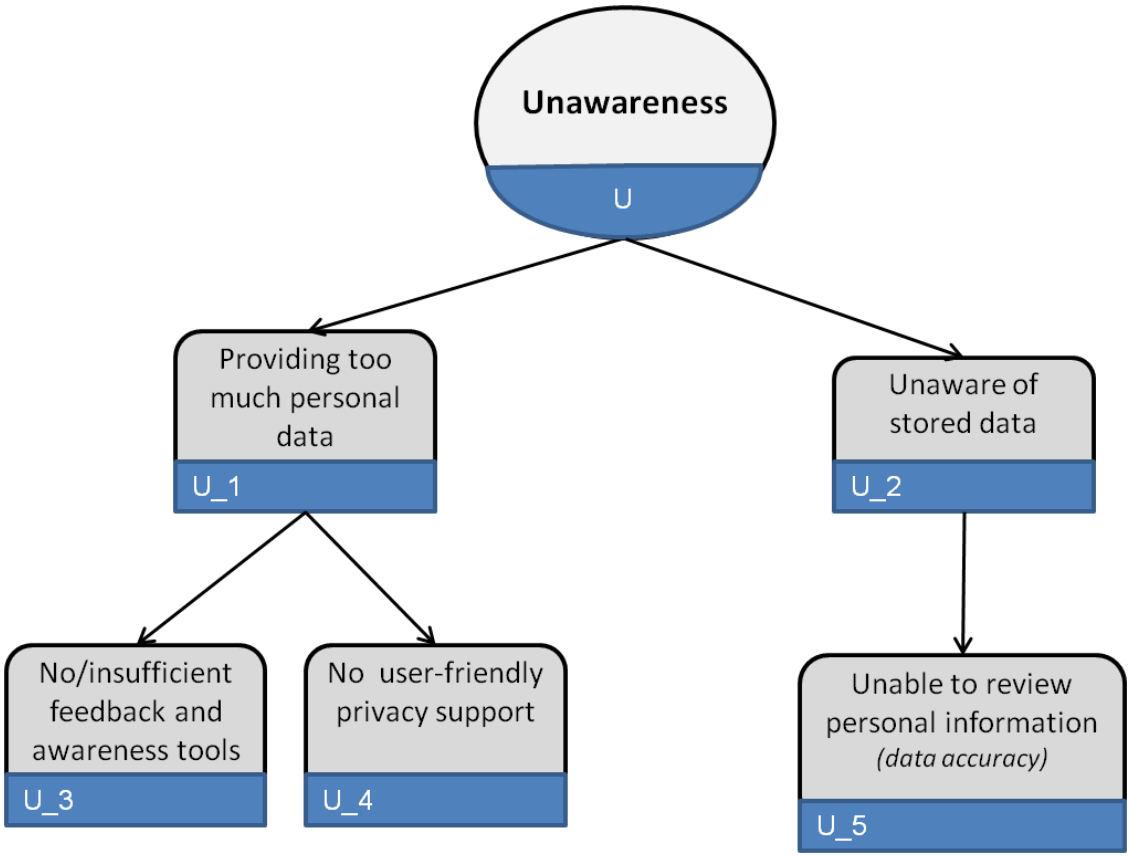
Consequences

- *Linkability/ identifiability*: the more information is available, the easier it can be linked (and identified)

Impacted by

- /

Unawareness of entity



Tree in general

The user provides more personal identifiable information than required (about himself or another data subject), which has a negative influence on all the hard privacy objectives

Leaf nodes explanation

Providing too much information (U_1)

Unawareness often means the **user provides too much (personal) information** ***(U_1)***. This is specifically problematic when this information is identifiable, or when this information when combined with the already available data becomes identifiable. This threat thus closely relates to the identifiability threats concerning data minimization.

Currently some privacy-friendly techniques exist to assist the user in making aware decisions concerning the sharing of his data. When these techniques are not employed, the awareness is obviously threatened. Feedback and awareness tools (feedback tools that improve the user's understanding of privacy implications [1], or tools that visualize the user's data like the IdentityMirror [2] and privacy mirrors [3]) provide feedback on the data the user wants to share and presents its results when combined with already available (online) information. Also several types of nudges have been proposed for social networks to encourage the user to reflect on the information he wants to share [4] [5]: sentiment nudges that inform the user how a certain message might be perceived, picture nudges that show profile pictures of

(some) users that will be able to see the post, etc.
Facebook already provides some privacy feedback as it allows the user to access his timeline with the access control settings of a specific user. This raises awareness and can help prevent the oversharing of information.
When **no feedback and awareness tools*(U_3)*** are used, the user is likely not aware of the information (or its impact) he is sharing.

Also, **privacy support should be user-friendly *(U_4)***. For example, default settings (e.g. facebook settings) should be privacy-friendly**.** It should be prevented that information is automatically shared with many parties, often without knowledge of the user. Privacy-friendly settings should limit the exposure of (personal) data.
Also, in order for users to modify privacy settings according to their needs, the provided tools should be easy to use. The privacy configuration (e.g. Facebook privacy settings) should be easy to access and manage and should be represented in an understandable fashion.

Data accuracy (U_2)

Often data subjects are unaware of what data a system has actually collected and stored about him. A data subject should thus always have the possibility to **review his own data** (i.e. data that has been collected about him)[6].

[1] Lederer, s., Hong, J., Dey, A., Landay, J.: Personal Privacy through Understanding and Action: Five Pitfalls for Designers. Personal and Ubiquitous Computing 8, 440--454 (2004)

[2] Liu, H., Maes, P., Davenport, G.: Unraveling the Taste Fabric of Social Networks. International Journal on Semantic Web and Information Systems (IJSWIS) 2(1)

[3] Nguyen, D., Mynatt, E.: Privacy Mirrors: Understanding and Shaping Socio-technical Ubiquitous Computing Systems. (2001)

[4] Yang Wang, Pedro Giovanni Leon, Kevin Scott, Xiaoxuan Chen, Alessandro Acquisti, and Lorrie Faith Cranor. 2013. Privacy nudges for social media: an exploratory Facebook study. In Proceedings of the 22nd international conference on World Wide Web companion (WWW '13 Companion). 763-770.

[5] Presentation by Lorrie Cranor on Privacy Nudges and Self-Censorship on Social Media (summarizing [5])

[6] Note that a data subject should even be able to request updates of the data if certain information is no longer applicable or correct. This is however included in the *Non-Repudiation threat tree (NR_ds7)* as it not specific to user awareness.

Non-compliance

In general

Not being compliant with legislation, regulations, and corporate policies.

Consequences

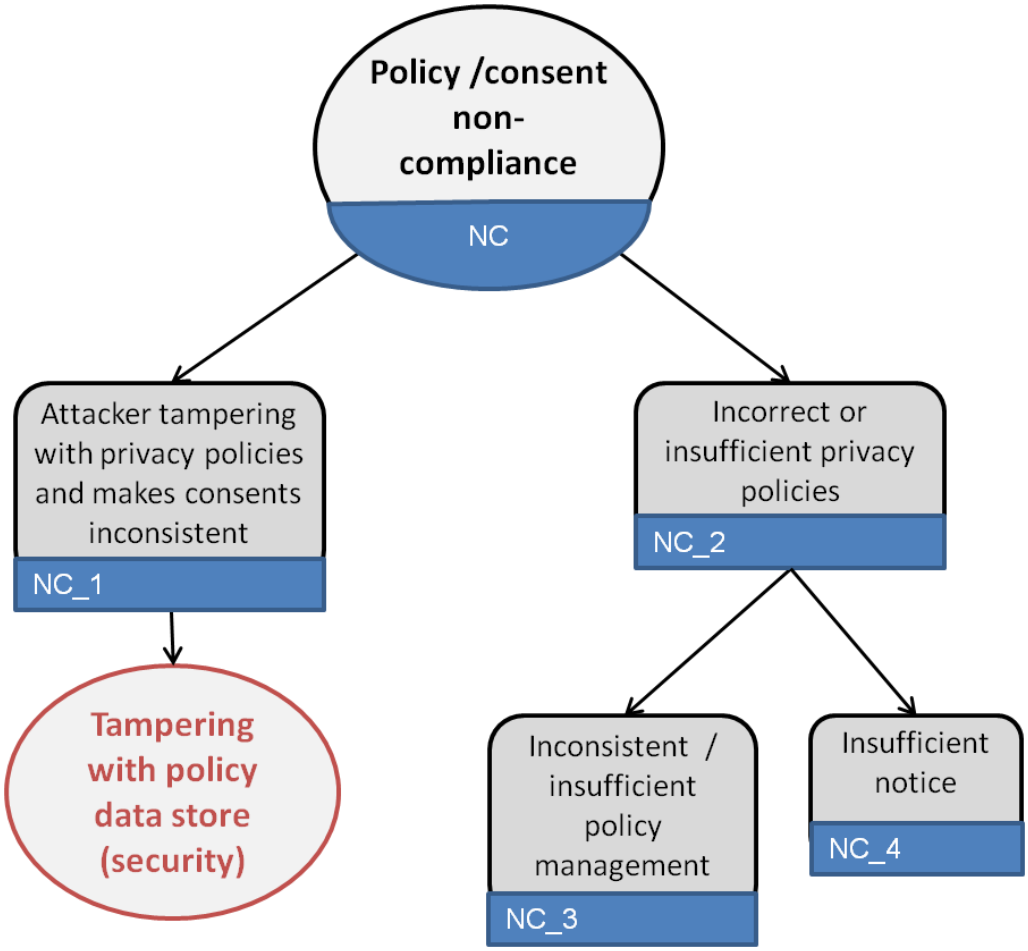
- Can lead to fines (when violating legislation, or not adhering to the communicated corporate policies)
- Can lead to loss of image, credibility, etc.

Impacted by

- Security officer/legal audit/… : a person responsible for the system's compliance (+)
- Tampering with the policy data store: when the policy database is not tamper-proof, attackers can alter the access control policies and user consents of the system (-)

Non-Compliance

No distinction is made between the different DFD elements, as compliance generally applies to the system as a whole.



Tree in general

A system is compliant when it adheres to its communicated corporate policies, to all (data protection) legislation, and to the user's consents.

The tree only aims at identifying compliance issues (related to privacy) from a high-level perspective. **To ensure full compliance, we advice the assistance of a legal expert.**

Leaf nodes explanation

Legal compliance is a very complicated domain and will generally require the assistance of a legal expert. Related work exists that summarizes the most common principles in data protection legislation.
Anton, Earp, and Reese [1] summarize 5 privacy protection goals as notice/awareness, choice/consent, access/participation, integrity/security, enforcement/redress.

Guarda and Zannone present 9 privacy principles [2]: fair and lawful processing, consent, purpose specification, minimality, minimal disclosure, information quality, data subject control, sensitivity, and information security.

Privacy policies can have several origins. They can be imposed by **law** (like data protection legislation), but also each company has its own **corporate** policies (which are also communicated with the users) to which the system should adhere. On top of that, the users themselves can also be involved in the privacy rules by means of **consents**. Users should be able to decide what happens with their data and who has access to their information. They can elicit their rules in the form of user consents. These consents can either be to allow certain actions to their personal information. Examples are: allow data to be used for research, or allow data to be communicated with a third party, or even allow person X to access (and/or update) my personal data). A consent can also restrict access, when a user decides he does not want a certain person or party to access his personal information (e.g. prohibit your neighbor who works in a hospital to access your medical records). Consents and privacy policies in general can, and should, be integrated into the system's access control policies as much as possible.

Attacker tampering with privacy policies and makes consents inconsistent (NC_1)

When the privacy policies and consents are integrated in the access control system, it is important that they are stored in a correct and consistent fashion. Clearly, when an attacker can **tamper with the policies** ***(NC_1)***, the attacker can alter or remove the policies (and consents) and the system can no longer ensure compliance. This can happen when **the database that stores the policies is susceptible to tampering threats** ***(T_DS of STRIDE)***.

Incorrect or insufficient privacy policies (NC_2)

To "automate" compliance, these policies need to be enforced in the system. When the **privacy policies are incorrectly or insufficiently implemented** ***(NC_2)***, the system will not be compliant. When **insufficient policy management** ***(NC_3)*** is provided, the system will not be compliant to the user consent requirements. Insufficient policy management exists when the system does not provide (user-friendly) support to the user to create or update user consents, or, when the created user consents are not correctly enforced in the system. This insufficient policy management can be both accidental and intentional.

Finally, also related to the system's corporate policies is the threat related to **notice** ***(NC_4)*** Clear, transparent notice of the applied policies should be provided to all users to inform them about the data that will be collected, stored, and processed [3].

Note that these threats are only related to privacy policies. Data protection compliance spans a much broad range of threats (most of which also occur in other LINDDUN threat trees). Those threats are mainly related to storage and overall management of data that is not compliant with legislation or the specified corporate policies. They include (but are not

limited to): insufficient minimization (collecting and storing too much information with respect to its purpose), storing data too long (data retention), storing sensitive data without the necessary precautions (anonymizing, encrypting,...), not providing the data subjects with access to their data, not respecting the 'right to be forgotten', ...
To guarantee full legal compliance, we advice the assistance of a legal expert.

[1] Anton, A.I.; Earp, J.B.; Reese, A., 2002, "Analyzing Website privacy requirements using a privacy goal taxonomy," Requirements Engineering, 2002. Proceedings. IEEE Joint International Conference on , pp.23,31

[2] Paolo Guarda and Nicola Zannone. 2009. "Towards the development of privacy-aware systems".Inf. Softw. Technol. 51, 2 (February 2009),pp. 337-350.

[3] Tools like Privacy Bird have already emerged that inform users about how information they provide to websites could be used. These tools interpret the website's privacy policies and translate them to information that is easier to grasp (as privacy policies are hardly ever read by users because they tend to be very extensive and hard to comprehend).
Ideally however, each system provides its own easy to grasp notice.